String Languages Generated by Total Deterministic Macro Tree Transducers*

Sebastian Maneth

Department of Computer Science, Leiden University, PO Box 9512,
2300 RA Leiden, The Netherlands, E-mail: maneth@wi.leidenuniv.nl

Abstract. The class of string languages obtained by taking the yields of output tree languages of total deterministic macro tree transducers (MTTs) is investigated. The first main result is that MTTs which are linear and nondeleting in the parameters generate the same class of string languages as total deterministic top-down tree transducers. The second main result is a so called "bridge theorem"; it can be used to show that there is a string language generated by a nondeterministic top-down tree transducer with monadic input, i.e., an ET0L language, which cannot be generated by an MTT. In fact, it is shown that this language cannot even be generated by the composition closure of MTTs; hence it is also not in the IO-hierarchy.

1 Introduction

Macro tree transducers [Eng80, CF82, EV85, EM98] are a well-known model of syntax-directed semantics (for a recent survey, see [FV98]). They are obtained by combining top-down tree transducers with macro grammars. In contrast to top-down tree transducers they have the ability to handle context information. This is done by parameters.

A total deterministic macro tree transducer (for short, MTT) M realizes a translation τ_M which is a function from trees to trees. The input trees may, for instance, be derivation trees of a context-free grammar which describes the syntax of some programming language (the source language). To every input tree s (viz. the derivation tree of a source program P) M associates the tree $\tau_M(s)$. This tree may then be interpreted in an appropriate semantic domain, e.g., yielding a program in another programming language (the target language): the semantics of P. One specific, quite popular, such domain is the one of strings with concatenation as only operation. More precisely, every symbol of rank greater than zero is interpreted as concatenation and constant symbols are interpreted as letters. The interpretation of a tree t in this domain is simply its *yield* (or frontier, i.e., the string obtained from t by reading its leaves from left to right). Thus, an MTT M can be seen as a translation device from trees to strings. Taking a tree language as input it generates a formal language as output. It

* This work was supported by the EC TMR Network GETGRATS.

W. Thomas (Ed.): FOSSACS'99, LNCS 1578, pp. 258-272, 1999.

is this class of formal languages (viz. the sets of target programs that can be generated) which we investigate in this paper.

An MTT M such that each right-hand side of a rule is linear and nondeleting in the parameters, that is, every parameter occurs exactly once, will be called *simple in the parameters*. This means that M cannot copy by means of its parameters. We prove that the class of string languages generated by such MTTs equals the class of string languages generated by top-down tree transducers. Hence the parameters can be eliminated. It is known that for unrestricted MTTs this is not the case; also if we consider output tree languages, MTTs that are simple in the parameters can do more than top-down tree transducers: they can generate tree languages that have non-regular path languages, which cannot be done by top-down tree transducers. For a more severe restriction, namely, the finite copying restriction, MTTs generate the same class of string languages as finite copying top-down tree transducers (Corollary 7.10 of [EM98]).

Now consider the case that we want to prove that a certain tree language R cannot be generated (as output tree language) by any MTT. In general this is difficult for there are very few appropriate tools: there exists a pumping lemma [Küh98] for a restricted case of MTTs. If we know that the string language obtained by taking the yields of the trees in R cannot be generated by any MTT, then we immediately know that R cannot be generated by an MTT. Since there are many tree languages with the same yield language, it is much stronger to know that a string language cannot be generated by an MTT than to know this for a tree language. We present a tool which is capable of proving that certain string languages L cannot be generated by an MTT. More precisely we will show that if L is of the form $f(L')$ for some fixed operation f, then L' can be generated by an MTT which is simple in the parameters; by our first result this means that L' can be generated by a top-down tree transducer. The proof is a direct generalization of Fischer's result on IO macro grammars: in the proof of Theorem 3.4.3 in [Fis68] it is proved that if $f(L)$ is an IO macro language then L can be generated by an IO macro grammar which is simple in the parameters. The result shows that the structure of L forces it from a bigger into a smaller class; it gives a "bridge" from the bigger (viz. unrestricted MTTs) into the smaller class (viz. MTTs which are simple in the parameters). For this smaller class, i.e., the class of string languages generated by top-down tree transducers, there exists another bridge theorem into yet another smaller class (using the same operation f), namely the class of string languages generated by finite copying top-down tree transducers [ERS80]. Due to the limited copying power of this class, it only contains languages that are of linear growth (they have the "Parikh property"); thus, languages like $L_{\text{exp}} = \{a^{2^n} \mid a \geq 0\}$ are not in this class. Altogether we get that $f(f(L'))$, where L' is a non-Parikh language (e.g., L_{exp}) cannot be generated by an MTT; in fact, we prove that it cannot be generated by any composition of MTTs.

This paper is structured as follows. In Section 2 we fix some notions used throughout the paper. Section 3 recalls macro tree transducers. In Section 4 we establish our two main results. Section 5 concludes with some open problems.

2 Preliminaries

The set $\{0, 1, \dots\}$ of natural numbers is denoted by $\mathbb{N}$. The empty set is denoted by $\varnothing$. For $k \in \mathbb{N}$, $[k]$ denotes the set $\{1, \dots, k\}$; thus $[0] = \varnothing$. For a set A, A^* is the set of all strings over A. The empty string is denoted by ε and the length of a string w is denoted $|w|$. For strings $v, w_1, \dots, w_n \in A^*$ and distinct $a_1, \dots, a_n \in A$, we denote by $v[a_1 \leftarrow w_1, \dots, a_n \leftarrow w_n]$ the result of (simultaneously) substituting w_i for every occurrence of a_i in v. Note that $[a_1 \leftarrow w_1, \dots, a_n \leftarrow w_n]$ is a homomorphism on strings. For a condition P on a and w we use, similar to set notation, $[a \leftarrow w \mid P]$ to denote the substitution $[L]$, where L is the list of all $a \leftarrow w$ for which condition P holds.

For functions $f: A \to B$ and $g: B \to C$ their composition is $(f \circ g)(x) = g(f(x))$; note that the order of f and g is nonstandard. For sets of functions F and G their composition is $F \circ G = \{f \circ g \mid f \in F, g \in G\}$.

2.1 Trees

A set Σ together with a mapping $\mathrm{rank}_\Sigma: \Sigma \to \mathbb{N}$ is called a *ranked* set. For $k \in \mathbb{N}$, $\Sigma^{(k)}$ denotes the set $\{\sigma \in \Sigma \mid \mathrm{rank}_\Sigma(\sigma) = k\}$. We will often write $\sigma^{(k)}$ to indicate that $\mathrm{rank}_\Sigma(\sigma) = k$.

The *set of trees over* Σ, denoted by T_Σ, is the smallest set of strings $T \subseteq (\Sigma \cup \{(,), ,\})^*$ such that if $\sigma \in \Sigma^{(k)}$, $k \geq 0$, and $t_1, \dots, t_k \in T$, then $\sigma(t_1, \dots, t_k) \in T$. For $\alpha \in \Sigma^{(0)}$ we denote the tree $\alpha()$ also by α. For a set A, $T_\Sigma(A)$ denotes the set $T_{\Sigma \cup A}$, where every symbol of A has rank 0 and $\langle \Sigma, A \rangle$ denotes the ranked set $\{\langle \sigma, a \rangle^{(k)} \mid \sigma \in \Sigma^{(k)}, a \in A\}$ (if Σ is unranked, then every symbol in $\langle \Sigma, A \rangle$ is of rank zero). We fix the set of *variables* X as $\{x_1, x_2, \dots\}$ and the set of *parameters* Y as $\{y_1, y_2, \dots\}$. For $k \in \mathbb{N}$, X_k and Y_k denote the sets $\{x_1, \dots, x_k\}$ and $\{y_1, \dots, y_k\}$, respectively.

For a tree t, the string obtained by reading the labels of its leaves from left to right, called the *yield of* t, is denoted by yt. The special symbol e of rank zero will be used to denote the empty string ε (e.g., $y(\sigma(a, e)) = a$ and $ye = \varepsilon$). For a string $w = a_1 \cdots a_n$ and a binary symbol b let $\mathrm{comb}_b(w)$ denote the tree $b(a_1, b(a_2, \dots b(a_n, e) \dots))$ over $\{b^{(2)}, a_1^{(0)}, \dots, a_n^{(0)}\}$; note that $y\mathrm{comb}_b(w) = w$.

A subset L of T_Σ is called a *tree language*. The class of all *regular* (or, *recognizable*) tree languages is denoted by $REGT$ (cf., e.g., [GS97]). For a tree language L we denote by yL the string language $\{yt \mid t \in L\}$ and for a class of tree languages $\mathcal{L}$ we denote by $y\mathcal{L}$ the class of string languages $\{yL \mid L \in \mathcal{L}\}$. A relation $\tau \subseteq T_\Sigma \times T_\Delta$ is called a *tree translation* or simply translation; by $y\tau$ we denote $\{(s, yt) \mid (s, t) \in \tau\}$. For a tree language $L \subseteq T_\Sigma$, $\tau(L)$ denotes the set $\{t \in T_\Delta \mid (s, t) \in \tau \text{ for some } s \in L\}$. For a class $\mathcal{T}$ of tree translations and a class $\mathcal{L}$ of tree languages, $\mathcal{T}(\mathcal{L})$ denotes the class of tree languages $\{\tau(L) \mid \tau \in \mathcal{T}, L \in \mathcal{L}\}$ and $y\mathcal{T}$ denotes $\{y\tau \mid \tau \in \mathcal{T}\}$.

2.2 Tree Substitution and Relabelings

Note that trees are particular strings and that string substitution as defined in the beginning of this section is applicable to a tree to replace symbols of rank zero; we refer to this type of substitution as "first order tree substitution".

Let Σ be a ranked set and let $\sigma_1, \ldots, \sigma_n$ be distinct elements of Σ, $n \geq 1$, and for each $i \in [n]$ let s_i be a tree in $T_\Sigma(Y_k)$, where $k = \mathrm{rank}_\Sigma(\sigma_i)$. For $t \in T_\Sigma$, the *second order substitution of s_i for σ_i in t*, denoted by $t[\![\sigma_1 \leftarrow s_1, \ldots, \sigma_n \leftarrow s_n]\!]$ is inductively defined as follows (abbreviating $[\![\sigma_1 \leftarrow s_1, \ldots, \sigma_n \leftarrow s_n]\!]$ by $[\![\ldots]\!]$). For $t = \sigma(t_1, \ldots, t_k)$ with $\sigma \in \Sigma^{(k)}$, $k \geq 0$, and $t_1, \ldots, t_k \in T_\Sigma$, (i) if $\sigma = \sigma_i$ for an $i \in [n]$, then $t[\![\ldots]\!] = s_i[y_j \leftarrow t_j[\![\ldots]\!] \mid j \in [k]]$ and (ii) otherwise $t[\![\ldots]\!] = \sigma(t_1[\![\ldots]\!], \ldots, t_k[\![\ldots]\!])$. For a condition P on σ and s, we use $[\![\sigma \leftarrow s \mid P]\!]$ to denote the substitution $[\![L]\!]$, where L is the list of all $\sigma \leftarrow s$ for which condition P holds.

A (deterministic) *finite state relabeling* M is a tuple $(Q, \Sigma, \Delta, F, R)$, where Q is a finite set of *states*, Σ and Δ are ranked alphabets of *input* and *output symbols*, respectively, $F \subseteq Q$ is a set of *final states*, and R is a finite set of rules such that for every $\sigma \in \Sigma^{(k)}$, $k \geq 0$, and $q_1, \ldots, q_k \in Q$, there is exactly one rule of the form $\sigma(\langle q_1, x_1\rangle, \ldots, \langle q_k, x_k\rangle) \to \langle q, \delta(x_1, \ldots, x_k)\rangle$ in R, where $q \in Q$ and $\delta \in \Delta^{(k)}$. The rules of M are used as term rewriting rules, and the rewrite relation induced by M (on $T_{\langle Q, T_\Delta\rangle \cup \Sigma}$) is denoted by $\Rightarrow_M$. The translation realized by M is $\tau_M = \{(s,t) \in T_\Sigma \times T_\Delta \mid s \Rightarrow_M^* \langle q, t\rangle, q \in F\}$. The class of all translations that can be realized by finite state relabelings is denoted by $DQRELAB$.

3 Macro Tree Transducers

A macro tree transducer is a syntax-directed translation device in which the translation of an input subtree may depend on its context. The context information is processed by parameters. We will consider total deterministic macro tree transducers only.

Definition 1. A *macro tree transducer* (for short, MTT) is a tuple $M = (Q, \Sigma, \Delta, q_0, R)$, where Q is a ranked alphabet of *states*, Σ and Δ are ranked alphabets of *input* and *output symbols*, respectively, $q_0 \in Q^{(0)}$ is the *initial state*, and R is a finite set of *rules*; for every $q \in Q^{(m)}$ and $\sigma \in \Sigma^{(k)}$ with $m, k \geq 0$ there is exactly one rule of the form $\langle q, \sigma(x_1, \ldots, x_k)\rangle(y_1, \ldots, y_m) \to \zeta$ in R, where $\zeta \in T_{\langle Q, X_k\rangle \cup \Delta}(Y_m)$.

A rule of the form $\langle q, \sigma(x_1, \ldots, x_k)\rangle(y_1, \ldots, y_m) \to \zeta$ is called the (q,σ)-rule and its right-hand side ζ is denoted by $\mathrm{rhs}_M(q,\sigma)$; it is also called a q-rule.

The rules of M are used as term rewriting rules and by $\Rightarrow_M$ we denote the *derivation relation induced by M* (on $T_{\langle Q, T_\Sigma\rangle \cup \Delta}(Y)$). The *translation realized by M*, denoted by τ_M is the total function $\{(s,t) \in T_\Sigma \times T_\Delta \mid \langle q_0, s\rangle \Rightarrow_M^* t\}$. The class of all translations that can be realized by MTTs is denoted by *MTT*. If for every $\sigma \in \Sigma$, $q \in Q^{(m)}$, $m \geq 0$, and $j \in [m]$, y_j occurs *exactly once* in $\mathrm{rhs}_M(q,\sigma)$ (i.e., the rules of M are linear and nondeleting in Y_m), then M is *simple in the parameters* (for short *sp*; we say, M is an $\mathrm{MTT_{sp}}$). The class of all translations that can be realized by $\mathrm{MTT_{sp}}$s is denoted by MTT_{sp}. If all states of an MTT are of rank zero, then M is called *top-down tree transducer*. The class of translations realized by top-down tree transducers is denoted by T. For top-down tree transducers we also consider the case that for a state q and an input symbol σ there may be more than one rule of the form $\langle q, \sigma(x_1, \ldots, x_k)\rangle \to \zeta$ in R. Such a top-down tree transducer is called *nondeterministic* and the corresponding class of translations is denoted by $N\text{-}T$ (note that this is a class of relations rather than total functions). The class of translations realized by nondeterministic top-down tree transducer with monadic input (i.e., each input symbol is of rank 0 or 1) is denoted by $N\text{-}T_{\text{mon}}$.

Let us now consider an example of an MTT.

Example 1. Let $M = (Q, \Sigma, \Sigma, q_0, R)$ be the MTT_{sp} with $Q = \{q^{(2)}, q_0^{(0)}\}$, $\Sigma = \{\sigma^{(2)}, a^{(0)}, b^{(0)}\}$, and R consisting of the following rules.

$$
\begin{array}{ll}
\langle q_0, \sigma(x_1, x_2)\rangle & \to \langle q, x_2\rangle(\langle q_0, x_1\rangle, \langle q_0, x_1\rangle) \\
\langle q, \sigma(x_1, x_2)\rangle(y_1, y_2) & \to \langle q, x_2\rangle(\sigma(y_1, \langle q_0, x_1\rangle), \sigma(\langle q_0, x_1\rangle, y_2)) \\
\langle q_0, a\rangle & \to a \\
\langle q, a\rangle(y_1, y_2) & \to \sigma(y_1, y_2) \\
\langle q_0, b\rangle & \to b \\
\langle q, b\rangle(y_1, y_2) & \to \sigma(y_2, y_1)
\end{array}
$$

Consider the input tree $t = \sigma(a, \sigma(b, \sigma(b, b)))$. Then a derivation by M looks as follows.

$$
\begin{aligned}
\langle q_0, t\rangle &\Rightarrow_M \langle q, \sigma(b, \sigma(b, b))\rangle(\langle q_0, a\rangle, \langle q_0, a\rangle) \\
&\Rightarrow_M^* \langle q, \sigma(b, \sigma(b, b))\rangle(a, a) \\
&\Rightarrow_M \langle q, \sigma(b, b)\rangle(\sigma(a, \langle q_0, b\rangle), \sigma(\langle q_0, b\rangle, a)) \\
&\Rightarrow_M^* \langle q, \sigma(b, b)\rangle(\sigma(a, b), \sigma(b, a)) \\
&\Rightarrow_M^* \langle q, b\rangle(\sigma(\sigma(a, b), b), \sigma(b, \sigma(b, a))) \\
&\Rightarrow_M \sigma(\sigma(b, \sigma(b, a)), \sigma(\sigma(a, b), b))
\end{aligned}
$$

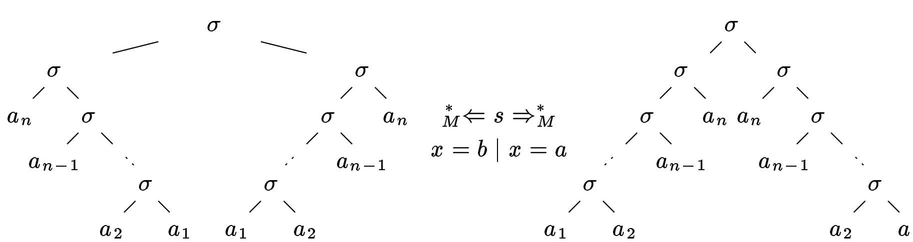

Fig. 1. Translations of M with input s for $x = b$ and $x = a$

In Fig. 1 it is shown how the translations for trees of the form

$$s = \sigma(a_1, \sigma(a_2, \ldots \sigma(a_n, x) \ldots))$$

with $a_1, \ldots, a_n \in \Sigma^{(0)}$ and $n \geq 1$ look like. If $x = a$ then $y\tau_M(s) = ww^r$ and if $x = b$ then $y\tau_M(s) = w^r w$, where $w = a_1 \cdots a_n$ and w^r denotes the reverse of w (i.e., the string $a_n a_{n-1} \cdots a_1$). Note that M is sp because both y_1 and y_2 appear exactly once in the right-hand side of each q-rule of M. $\square$

The next lemma will be used in proofs by induction on the structure of the input tree. Let $M = (Q, \Sigma, \Delta, q_0, R)$ be an MTT. For every $q \in Q^{(m)}$ and $s \in T_\Sigma$ let the *q-translation of s*, denoted by $M_q(s)$, be the unique tree $t \in T_\Delta(Y_m)$ such that $\langle q, s\rangle(y_1, \ldots, y_m) \Rightarrow_M^* t$. Note that, for $s \in T_\Sigma$, $\tau_M(s) = M_{q_0}(s)$. The q-translations of trees in T_Σ can be characterized inductively as follows.

Lemma 2. (cf. Definition 3.18 of [EV85]) *Let* $M = (Q, \Sigma, \Delta, q_0, R)$ *be an* MTT. *For every* $q \in Q$, $\sigma \in \Sigma^{(k)}$, $k \geq 0$, *and* $s_1, \ldots, s_k \in T_\Sigma$, $M_q(\sigma(s_1, \ldots, s_k)) = \mathrm{rhs}_M(q, \sigma)[\![\langle q', x_i\rangle \leftarrow M_{q'}(s_i) \mid \langle q', x_i\rangle \in \langle Q, X_k\rangle]\!]$.

4 String Languages Generated by MTT

To prove our first main result we need the following small lemma about second order tree substitution. It says that if we are considering the yield of a tree to which a second order tree substitution is applied, then inside the substitution merely the yields of the trees that are substituted are relevant.

Lemma 3. *Let* Σ *be a ranked alphabet,* $\gamma_1, \ldots, \gamma_n \in \Sigma$, *and* $t, s_1, s_1', \ldots, s_n, s_n' \in T_\Sigma(Y)$. *If* $ys_i = ys_i'$ *for every* $i \in [n]$, *then*

$$y(t[\![\gamma_1 \leftarrow s_1, \ldots, \gamma_n \leftarrow s_n]\!]) = y(t[\![\gamma_1 \leftarrow s_1', \ldots, \gamma_n \leftarrow s_n']\!]).$$

Lemma 3 can be proved by straightforward induction on t. We now show how to generate by a top-down tree transducer the string language generated by an $\mathrm{MTT_{sp}}$.

Lemma 4. $y\mathit{MTT}_{\mathrm{sp}} \subseteq y(\mathit{DQRELAB} \circ T)$.

Proof. Let $M = (Q, \Sigma, \Delta, q_0, R)$ be an $\mathrm{MTT_{sp}}$. We will construct a finite state relabeling N and a top-down tree transducer M' such that for every $s \in T_\Sigma$, $y(\tau_{M'}(\tau_N(s))) = y\tau_M(s)$. The idea is as follows. Let $q \in Q^{(m)}$ and $s \in T_\Sigma$. Then, since M is sp, $yM_q(s)$ is of the form

$$w = w_0 y_{j_1} w_1 y_{j_2} w_2 \cdots y_{j_m} w_m,$$

where $j_1, \ldots, j_m \in [m]$ are pairwise different and $w_0, \ldots, w_m \in (\Delta^{(0)})^*$. For a string of the form w (where the w_i are arbitrary strings not containing parameters) and for $0 \leq \nu \leq m$ we denote by $\mathrm{part}_\nu(w)$ the string w_ν. For every w_ν the top-down tree transducer M' has a state (q, ν) which computes w_ν. The information on the order of the parameters, i.e., the indices $j_1, \ldots, j_m$, will be determined by the finite state relabeling N in such a way that $\sigma \in \Sigma^{(k)}$ is relabeled by $(\sigma, (\mathrm{pos}_1, \ldots, \mathrm{pos}_k))$, where for each $i \in [k]$, pos_i is a mapping associating with every $q \in Q^{(m)}$ a bijection from $[m]$ to $[m]$. For instance, if s_i equals the tree s from above, then the σ in $\sigma(s_1, \ldots, s_i, \ldots, s_k)$ is relabeled by $(\sigma, (\mathrm{pos}_1, \ldots, \mathrm{pos}_k))$ and $\mathrm{pos}_i(q)(\nu) = j_\nu$ for $\nu \in [m]$. Formally, $N = (Q_N, \Sigma, \Gamma, Q_N, R_N)$, where

- Q_N is the set of all mappings pos which associate with every $q \in Q^{(m)}$ a bijection $\mathrm{pos}(q)$ from $[m]$ to $[m]$. For convenience we identify $\mathrm{pos}(q)$ with the string $j_1 \cdots j_m$ over $[m]$, where $\mathrm{pos}(q)(i) = j_i$ for $i \in [m]$.

- $\Gamma = \{(\sigma, (\text{pos}_1, \dots, \text{pos}_k))^{(k)} \mid \sigma \in \Sigma^{(k)}, k \geq 0, \text{pos}_1, \dots, \text{pos}_k \in Q_N\}$.
- For every $\sigma \in \Sigma^{(k)}$, $k \geq 0$, and $\text{pos}_1, \dots, \text{pos}_k \in Q_N$ let

$$\sigma(\langle \text{pos}_1, x_1 \rangle, \dots, \langle \text{pos}_k, x_k \rangle) \to \langle \text{pos}, (\sigma, (\text{pos}_1, \dots, \text{pos}_k))(x_1, \dots, x_k) \rangle$$

be in R_N, where for every $q \in Q^{(m)}$, $\text{pos}(q) = \text{order}(\text{rhs}_M(q, \sigma))$ and for $t \in T_{\langle Q, X_k \rangle \cup \Delta}(Y_m)$, $\text{order}(t)$ is the string over $[m]$ defined recursively as follows: if $t = y_j \in Y_m$, then $\text{order}(t) = j$, if $t = \delta(t_1, \dots, t_l)$ with $\delta \in \Delta^{(l)}$, $l \geq 0$, and $t_1, \dots, t_l \in T_{\langle Q, X_k \rangle \cup \Delta}(Y_m)$, then $\text{order}(t) = \text{order}(t_1) \cdots \text{order}(t_l)$, and if $t = \langle q', x_i \rangle (t_1, \dots, t_l)$ with $\langle q', x_i \rangle \in \langle Q, X_k \rangle^{(l)}$, $l \geq 0$, and $t_1, \dots, t_l \in T_{\langle Q, X_k \rangle \cup \Delta}(Y_m)$, then $\text{order}(t) = \text{order}(t_{\text{pos}_i(q')(1)}) \cdots \text{order}(t_{\text{pos}_i(q')(l)})$.

It is straightforward to show (by induction on the structure of s) that N is defined in such a way that if $\tau_N(\sigma(s_1, \dots, s_k)) = (\sigma, (\text{pos}_1, \dots, \text{pos}_k))(\tilde{s}_1, \dots, \tilde{s}_k)$, then for every $i \in [k]$ and $q \in Q^{(m)}$,

$$yM_q(s_i) = w_0 y_{\text{pos}_i(q)(1)} w_1 y_{\text{pos}_i(q)(2)} w_2 \cdots y_{\text{pos}_i(q)(m)} w_m,$$

for some $w_0, \dots, w_m \in (\Delta^{(0)})^*$. In the induction step it can be shown that for $t \in T_{\langle Q, X_k \rangle \cup \Delta}(Y_m)$, $\text{order}(t) = j_1 \cdots j_m$, where $j_1, \dots, j_m \in [m]$, $yt[\![\dots]\!] = w_0 y_{j_1} w_1 y_{j_2} w_2 \cdots y_{j_m} w_m$ for some $w_0, \dots, w_m \in (\Delta^{(0)})^*$, and $[\![\dots]\!] = [\![\langle q', x_i \rangle \leftarrow M_{q'}(s_i) \mid \langle q', x_i \rangle \in \langle Q, X_k \rangle]\!]$.

We now define the top-down tree transducer $M' = (Q', \Gamma, \Delta', (q_0, 0), R')$, where

- $Q' = \{(q, \nu)^{(0)} \mid q \in Q^{(m)}, 0 \leq \nu \leq m\}$,
- $\Delta' = \Delta^{(0)} \cup \{b^{(2)}, e^{(0)}\}$, where $e \notin \Delta$, and
- for every $(q, \nu) \in Q'$, $(\sigma, (\text{pos}_1, \dots, \text{pos}_k)) \in \Gamma^{(k)}$, and $k \geq 0$ let the rule

$$\langle (q, \nu), (\sigma, (\text{pos}_1, \dots, \text{pos}_k))(x_1, \dots, x_k) \rangle \to \zeta$$

be in R', where $\zeta = \text{comb}_b(\text{part}_\nu(y(\xi[\![\_]\!])))$, $\xi = \text{rhs}_M(q, \sigma)$, and $[\![\_]\!]$ is the substitution

$$[\![\langle q', x_i \rangle \leftarrow \text{comb}_b(\langle (q', 0), x_i \rangle y_{\text{pos}_i(q')(1)} \langle (q', 1), x_i \rangle y_{\text{pos}_i(q')(2)} \cdots \\ y_{\text{pos}_i(q')(m)} \langle (q', m), x_i \rangle) \mid \langle q', x_i \rangle \in \langle Q, X_k \rangle^{(m)}]\!].$$

We now prove the correctness of M', i.e., that for every $s \in T_\Sigma$, $y(\tau_{M'}(\tau_N(s))) = y\tau_M(s)$. It follows from the next claim by taking $(q, \nu) = (q_0, 0)$.

Claim: For every $(q, \nu) \in Q'$ and $s \in T_\Sigma$, $y(M'_{(q,\nu)}(\tau_N(s))) = \text{part}_\nu(yM_q(s))$.

The proof of this claim is done by induction on the structure of s. Let $s = \sigma(s_1, \dots, s_k)$, $\sigma \in \Sigma^{(k)}$, $k \geq 0$, and $s_1, \dots, s_k \in T_\Sigma$. Then $y(M'_{(q,\nu)}(\tau_N(s))) = y(M'_{(q,\nu)}((\sigma, (\text{pos}_1, \dots, \text{pos}_k))(\tilde{s}_1, \dots, \tilde{s}_k)))$, where $\tilde{s}_i = \tau_N(s_i)$ for all $i \in [k]$. This equals $y(\zeta[\dots])$, where $\zeta = \text{rhs}_{M'}((q, \nu), (\sigma, (\text{pos}_1, \dots, \text{pos}_k)))$ and $[\dots] = [\langle (q', \nu'), x_i \rangle \leftarrow M'_{(q',\nu')}(\tau_N(s_i)) \mid \langle (q', \nu'), x_i \rangle \in \langle Q', X_k \rangle]$. By the definition of the rules of M', $\zeta = \text{comb}_b(\text{part}_\nu(y(\xi[\![\_]\!])))$, where $\xi = \text{rhs}_M(q, \sigma)$ and $[\![\_]\!]$ is as above. By applying y (yield) and the induction hypothesis we get $\text{part}_\nu(y(\xi[\![\_]\!]))\Psi$, where Ψ is the string substitution $[\langle(q',\nu'),x_i\rangle \leftarrow \mathrm{part}_{\nu'}(yM_{q'}(s_i)) \mid \langle(q',\nu'),x_i\rangle\in\langle Q',X_k\rangle]$. Since Ψ does not change parameters, we can move it inside the application of part_ν to get $\mathrm{part}_\nu(y(\xi[\![\_]\!])\Psi)$. If we move Ψ inside the application of y (yield) we get $\mathrm{part}_\nu(y(\xi[\![\_]\!]\Psi'))$, where Ψ' denotes the first order tree substitution of replacing $\langle(q',\nu'),x_i\rangle$ of rank zero by a tree with $\mathrm{part}_{\nu'}(yM_{q'}(s_i))$ as yield. Applying Ψ' inside of $[\![\_]\!]$ amounts to replacing $\langle q',x_i\rangle$ by a tree with yield $w = \mathrm{part}_0(yM_{q'}(s_i))y_{\mathrm{pos}_i(q')(1)}\cdots\mathrm{part}_m(yM_{q'}(s_i))$. By the correctness of the finite state relabeling N, $w = yM_{q'}(s_i)$. Since, by Lemma 3, we can put any tree with yield w in the second order substitution, taking $M_{q'}(s_i)$ we get $\mathrm{part}_\nu(y(\xi[\![\ldots]\!]))$ with $[\![\ldots]\!] = [\![\langle q',x_i\rangle \leftarrow M_{q'}(s_i) \mid \langle q',x_i\rangle \in \langle Q,X_k\rangle]\!]$. By Lemma 2 this is equal to $\mathrm{part}_\nu(yM_q(s))$ which ends the proof of the claim. $\square$

Let us look at an example of an application of the construction in the proof of Lemma 4.

Example 2. Let M be the $\mathrm{MTT}_{\mathrm{sp}}$ of Example 1. We construct the finite state relabeling N and the top-down tree transducer M' following the construction in the proof of Lemma 4. Let $N = (Q_N,\Sigma,\Gamma,Q_N,R_N)$ be the finite state relabeling with $Q_N = \{q_{12},q_{21}\}$, $q_{12} = \{(q_0,\varepsilon),(q,12)\}$, $q_{21} = \{(q_0,\varepsilon),(q,21)\}$, and $\Gamma = \{(\sigma,(q_{12},q_{12}))^{(2)}, (\sigma,(q_{12},q_{21}))^{(2)}, (\sigma,(q_{21},q_{12}))^{(2)}, (\sigma,(q_{21},q_{21}))^{(2)}, (a,())^{(0)}, (b,())^{(0)}\}$. The set R_N of rules of N consists of the rules

$$\begin{aligned} a &\to \langle q_{12},(a,())\rangle\\ b &\to \langle q_{21},(b,())\rangle\\ \sigma(\langle r,x_1\rangle,\langle r',x_2\rangle) &\to \langle r',(\sigma,(r,r'))(x_1,x_2)\rangle \qquad \text{for all } r,r'\in Q_N. \end{aligned}$$

Consider the tree $t = \sigma(a,\sigma(b,\sigma(b,b)))$ again. Then $\tau_N(t)$ equals

$$(\sigma,(q_{12},q_{21}))((a,()),(\sigma,(q_{21},q_{21}))((b,()),(\sigma,(q_{21},q_{21}))((b,()),(b,()))))). \qquad (*)$$

We now construct the top-down tree transducer M'. Let $M' = (Q',\Gamma,\Delta',(q_0,0),R')$ with $Q' = \{(q_0,0)^{(0)},(q,0)^{(0)},(q,1)^{(0)},(q,2)^{(0)}\}$ and $\Delta' = \Sigma^{(0)}\cup\{b^{(2)},e^{(0)}\}$. For simplicity we write down the rules of M' as tree-to-string rules, i.e., we merely show the yield of the corresponding right-hand side. Let us consider in detail how to obtain the right-hand sides of the $((q,\nu),(\sigma,(r,q_{21})))$-rules for $0\le\nu\le 2$ and $r\in Q_N$. Since we are only interested in the yields, we have to consider the string $v = y(\mathrm{rhs}_M(q,\sigma)[\![\_]\!])$, where $[\![\_]\!]$ is defined as in the proof of Lemma 4. This string equals

$$\underbrace{\langle(q,0),x_2\rangle\langle(q_0,0),x_1\rangle}_{\mathrm{part}_0(v)}\, y_2\, \underbrace{\langle(q,1),x_2\rangle}_{\mathrm{part}_1(v)}\, y_1\, \underbrace{\langle(q_0,0),x_1\rangle\langle(q,2),x_2\rangle}_{\mathrm{part}_2(v)}.$$

Hence, for every $r\in Q_N$ and $0\le\nu\le 2$, $y\mathrm{rhs}_{M'}((q,\nu),(\sigma,(r,q_{21}))) = \mathrm{part}_\nu(v)$; similarly we get $y\mathrm{rhs}_{M'}((q,0),(\sigma,(r,q_{12}))) = \langle(q,0),x_2\rangle$,

$$\begin{aligned} y\mathrm{rhs}_{M'}((q,1),(\sigma,(r,q_{12}))) &= \langle(q_0,0),x_1\rangle\langle(q,1),x_2\rangle\langle(q_0,0),x_1\rangle,\\ y\mathrm{rhs}_{M'}((q,2),(\sigma,(r,q_{12}))) &= \langle(q,2),x_2\rangle. \end{aligned}$$

The remaining rules are, for $0\le\nu\le 2$ and $r,r'\in Q_N$,

$$\begin{aligned}
\langle (q_0,0),(\sigma,(r,r'))(x_1,x_2)\rangle &\to \langle (q,0),x_2\rangle\langle (q_0,0),x_1\rangle\langle (q,1),x_2\rangle \\
&\qquad\qquad\qquad \langle (q_0,0),x_1\rangle\langle (q,2),x_2\rangle \\
\langle (q_0,0),(a,())\rangle &\to a \\
\langle (q_0,0),(b,())\rangle &\to b \\
\langle (q,\nu),(a,())\rangle &\to \varepsilon \\
\langle (q,\nu),(b,())\rangle &\to \varepsilon
\end{aligned}$$

Finally, consider the derivation by M' with input tree $t' = \tau_N(t)$ (shown in $(*)$). Denote by $t'/2$ the tree $\tau_N(\sigma(b,\sigma(b,b)))$ and by $t'/22$ the tree $\tau_N(\sigma(b,b))$. Again we merely show the corresponding yields.

$$\begin{aligned}
& \langle (q_0,0),t'\rangle \\
\Rightarrow_{M'}\ & \langle (q,0),t'/2\rangle\langle (q_0,0),(a,())\rangle\langle (q,1),t'/2\rangle\langle (q_0,0),(a,())\rangle\langle (q,2),t'/2\rangle \\
\Rightarrow^*_{M'}\ & \langle (q,0),t'/22\rangle\langle (q_0,0),(b,())\rangle\, a\, \langle (q,1),t'/22\rangle\, a\, \langle (q_0,0),(b,())\rangle\langle (q,2),t'/22\rangle \\
\Rightarrow^*_{M'}\ & \langle (q,0),(b,())\rangle\, bba\, \langle (q,1),(b,())\rangle\, abb\, \langle (q,2),(b,())\rangle \\
\Rightarrow^*_{M'}\ & bbaabb.
\end{aligned}$$
□

From Lemma 4 we obtain our first main result: $\mathrm{MTT_{sp}}$s and top-down tree transducers generate the same class of string languages if they take as input a class of tree languages that is closed under finite state relabelings.

Theorem 5. *Let $\mathcal{L}$ be a class of tree languages that is closed under finite state relabelings. Then $yMTT_{\mathrm{sp}}(\mathcal{L}) = yT(\mathcal{L})$.*

Proof. By Lemma 4, $yMTT_{\mathrm{sp}}(\mathcal{L}) \subseteq yT(\mathcal{L})$ and since every top-down tree transducer is an $\mathrm{MTT_{sp}}$, $yT(\mathcal{L}) \subseteq yMTT_{\mathrm{sp}}(\mathcal{L})$. □

Since the class $REGT$ of regular tree languages is closed under finite state relabelings (cf. Proposition 20.2 of [GS97]), we get $yMTT_{\mathrm{sp}}(REGT) = yT(REGT)$ from Theorem 5. For top-down tree transducers it is known (Theorem 3.2.1 of [ERS80] and Theorem 4.3 of [Man98b]) that $T(REGT)$ is equal to the class $OUT(T)$ of output tree languages of top-down tree transducers (i.e., taking the particular regular tree language T_Σ as input). In fact, it is shown in [Man98b] that for any class Ψ of tree translations which is closed under left composition with "semi-relabelings", which are particular linear top-down tree translations, $\Psi(REGT) = OUT(\Psi)$. Since it can be shown that MTT_{sp} is closed under left composition with top-down tree translations we get that $yOUT(MTT_{\mathrm{sp}}) = yOUT(T)$, i.e., $\mathrm{MTT_{sp}}$s and top-down tree transducers generate the same class of string languages. If we consider $\mathrm{MTT_{sp}}$s with monadic output alphabet, then the class of path languages generated by them taking regular tree languages as input is also equal to $yT(REGT)$ (cf. the proof of Lemma 7.6 of [EM98]). Thus, the classes of path and string languages generated by $\mathrm{MTT_{sp}}$s are equal.

We now move to our second main result. First we define the operation $\mathrm{rub}_{b_1,\dots,b_n}$ which inserts the symbols $b_1,\dots,b_n$ ("*rub*bish") anywhere in the strings of the language to which it is applied. Let A be an alphabet, $L \subseteq A^*$ a language, and $b_1,\dots,b_n$ new symbols not in A. Then $\mathrm{rub}_{b_1,\dots,b_n}(L)$ denotes the language

$$\{w_1a_1w_2a_2\cdots w_ka_kw_{k+1} \mid a_1\cdots a_k \in L, k \geq 1, w_1,\dots,w_{k+1} \in \{b_1,\dots,b_n\}^*\}.$$

The following theorem shows that if an MTT M generates $\mathrm{rub}_0(L)$ (where $\mathrm{rub}_0 = \mathrm{rub}_{b_1,\ldots,b_n}$ for $n = 1$ and $b_1 = 0$) then, due to the nondeterminism inherent in rub_0, M cannot make use of its copying facility.

Theorem 6. *Let $\mathcal{L}$ be a class of tree languages which is closed under finite state relabelings and under intersection with regular tree languages, and let $L \subseteq A^*$. If $\mathrm{rub}_0(L) \in yMTT(\mathcal{L})$ then $L \in yMTT_{\mathrm{sp}}(\mathcal{L})$.*

Proof. Let $M = (Q, \Sigma, \Delta, q_0, R)$ be an MTT and $K \in \mathcal{L}$ such that $y\tau_M(K) = \mathrm{rub}_0(L)$ and $\Delta^{(0)} = A \cup \{0\}$. By Lemma 6.6 of [EM98] we may assume that M is nondeleting, i.e., for every $q \in Q^{(m)}$ and $j \in [m]$, y_j appears at least once in the right-hand side of each q-rule. Consider a string of the form

$$a_1 0^{n_1} a_2 0^{n_2} \cdots a_l 0^{n_l} a_{l+1}$$

with $l \geq 0$, $a_1, \ldots, a_{l+1} \in A$, and all $n_1, \ldots, n_l \geq 0$ pairwise different. We call such a string δ-string. Clearly, it is sufficient to consider only δ-strings in order to generate a tree language R with $yR = L$ (if we can construct an $\mathrm{MTT}_{\mathrm{sp}}$ which generates as yield language at least all δ-strings in $\mathrm{rub}_0(L)$, then by deletion of 0s we obtain an $\mathrm{MTT}_{\mathrm{sp}}$ which generates L as yield language). Consider the right-hand side of a rule of M in which some parameter y_j occurs more than once. If, during the derivation of a tree which has as yield a δ-string, this rule was applied, then the tree which is substituted for y_j in this derivation contains at most one symbol in A. Because otherwise, due to copying, the resulting string would not be a δ-string. Hence, when deriving a δ-string, a rule which contains multiple occurrences of a parameter y_j is only applicable if the yield of the tree being substituted for y_j contains at most one symbol in A. Based on this fact we can construct the $\mathrm{MTT}_{\mathrm{sp}}$ M' which generates L. The information whether the yield of the tree which will be substituted for a certain parameter contains none, one, or more than one occurrences of a symbol in A is determined by relabeling the input tree. Then this information is kept in the states of M'. More precisely, we will define a finite state relabeling N which relabels $\sigma \in \Sigma^{(k)}$ in the tree $\sigma(s_1, \ldots, s_k)$ by $(\sigma, (\phi_1, \ldots, \phi_k))$, where for every $i \in [k]$ and $q \in Q$,

$$\phi_i(q) = \begin{cases} e & \text{if } yM_q(s_i) \text{ contains no symbol in } A \\ a & \text{if } yM_q(s_i) = waw' \text{ with } w, w' \in (Y \cup \{0\})^* \\ dd & \text{otherwise,} \end{cases}$$

where $a \in A$ and d is an arbitrary symbol in A. Before we define N, let us define an auxiliary notion. For $w \in (\Delta^{(0)} \cup Y)^*$ let $\mathrm{oc}(w)$ be defined as follows. If $w \in (Y \cup \{0\})^*$, then $\mathrm{oc}(w) = e$; if $w = w_1 a w_2$ with $a \in A$ and $w_1, w_2 \in (Y \cup \{0\})^*$, then $\mathrm{oc}(w) = a$; and otherwise $\mathrm{oc}(w) = dd$.

Let $N = (Q_N, \Sigma, \Gamma, Q_N, R_N)$ be the finite state relabeling with

- $Q_N = \{\phi \mid \phi : Q \to (\{e, dd\} \cup A)\}$,
- $\Gamma = \{(\sigma, (\phi_1, \ldots, \phi_k))^{(k)} \mid \sigma \in \Sigma^{(k)}, k \geq 0, \phi_1, \ldots, \phi_k \in Q_N\}$, and

- R_N containing for every $\phi_1, \ldots, \phi_k \in Q_N$ and $\sigma \in \Sigma^{(k)}$ with $k \geq 0$ the rule

$$\sigma(\langle\phi_1, x_1\rangle, \ldots, \langle\phi_k, x_k\rangle) \to \langle\phi, (\sigma, (\phi_1, \ldots, \phi_k))(x_1, \ldots, x_k)\rangle,$$

  where for every $q \in Q$, $\phi(q) = \mathrm{oc}(y(\mathrm{rhs}_M(q, \sigma)\Theta))$ and Θ denotes the second order substitution (where b is an arbitrary binary symbol)

$$[\![\langle q', x_i\rangle \leftarrow \mathrm{comb}_b(\phi_i(q')y_1 \cdots y_m) \mid \langle q', x_i\rangle \in \langle Q, X_k\rangle^{(m)}, m \geq 0]\!].$$

It should be clear that N realizes the relabeling as described above.

We now define $M' = (Q', \Gamma, \Delta', q_0', R')$ to be the MTT with

- $Q' = \{(q, \varphi) \mid q \in Q^{(m)}, m \geq 0, \varphi : [m] \to (\{e, dd\} \cup A)\}$, where the rank of (q, φ) with $q \in Q^{(m)}$ is $|\{j \in [m] \mid \varphi(j) = dd\}|$,
- $\Delta' = (\Delta - \{0\}) \cup \{b^{(2)}, \mathrm{dummy}^{(2)}, e^{(0)}\}$, where b, dummy, and e are not in Δ,
- $q_0' = (q_0, \varnothing)$, and
- R' consisting of the following rules. For every $(q, \varphi) \in Q'^{(n)}$ and $(\sigma, (\phi_1, \ldots, \phi_k)) \in \Gamma^{(k)}$ with $n, k \geq 0$ and $q \in Q^{(m)}$ let

$$\langle(q, \varphi), (\sigma, (\phi_1, \ldots, \phi_k))(x_1, \ldots, x_k)\rangle(y_1, \ldots, y_n) \to \zeta$$

  be in R', where $\zeta = \mathrm{comb}_{\mathrm{dummy}}(y_1 \cdots y_n)$ if there is a $j \in [m]$ such that $\varphi(j) = dd$ and y_j occurs more than once in $t = \mathrm{rhs}_M(q, \sigma)$ and otherwise ζ is obtained from t by the following replacements:
  1. Replace each subtree $\langle q', x_i\rangle(t_1, \ldots, t_l)$ with $\langle q', x_i\rangle \in \langle Q, X_k\rangle^{(l)}$, $l \geq 0$, and $t_1, \ldots, t_l \in T_{\langle Q, X_k\rangle \cup \Delta}(Y_m)$, by the tree $\langle(q', \varphi'), x_i\rangle(t_{j_1}, \ldots, t_{j_{l'}})$, where $\{j_1, \ldots, j_{l'}\} = \varphi'^{-1}(dd)$ with $j_1 < \cdots < j_{l'}$ and for every $j \in [l]$, $\varphi'(j) = \mathrm{oc}(y(t_j\Theta\Psi))$ with Θ defined as above, and

$$\Psi = [y_\nu \leftarrow \varphi(\nu) \mid \nu \in [m]].$$

  2. For $j \in [m]$, replace y_j by $\varphi(j)$ if $\varphi(j) \neq dd$, and otherwise replace it by y_ν with $\nu = |\{\mu \mid \mu < j \text{ and } \varphi(\mu) = dd\}| + 1$.
  3. Replace each occurrence of 0 by e.

Obviously M' is sp. If we now consider the yields of all trees in $\tau_{M'}(\tau_N(K))$ which do not contain a dummy symbol, then we obtain L. By Theorem 7.4(1) of [EV85] $R = \tau_{M'}^{-1}(T_{\Delta' - \{\mathrm{dummy}\}})$ is a regular tree language. Hence $K' = \tau_N(K) \cap R$ is in $\mathcal{L}$ and $L = \tau_{M'}(K')$ is in $yMTT_{\mathrm{sp}}(\mathcal{L})$. □

Note that Theorems 5 and 6 can be applied to $\mathcal{L} = REGT$. Due to the next lemma they can also be applied to $\mathcal{L} = MTT^n(REGT)$ for $n \geq 1$.

Lemma 7. *Let $\mathcal{L}$ be a class of tree languages. If $\mathcal{L}$ is closed under finite state relabelings, then so is $MTT(\mathcal{L})$.*

Proof. Let $M = (Q, \Sigma, \Delta, q_0, R)$ be an MTT and let $N = (Q_N, \Delta, \Gamma, F, R_N)$ be a finite state relabeling. We now sketch how to construct a finite state relabeling N' and an MTT M' such that for every $s \in T_\Sigma$, $\tau_{M'}(\tau_{N'}(s)) = \tau_N(\tau_M(s))$. The idea is similar to the proof of Theorem 6. The relabeling N' replaces the symbol σ in $\sigma(s_1, \ldots, s_k) \in T_\Sigma$ by $(\sigma, (\phi_1, \ldots, \phi_k))$, where each ϕ_i associates with every $q \in Q^{(m)}$ a mapping of type $Q_N^m \to Q_N$ such that for $p_1, \ldots, p_m \in Q_N$, $\phi_i(q)(p_1, \ldots, p_m) = p$ if $M_q(s_i) \Rightarrow^*_{\hat{N}} \langle p, \tilde{s} \rangle$, where $\hat{N}$ is the extension of N to $\Delta \cup Y_M$ by rules $y_j \to \langle p_j, y_j \rangle$. Thus, if we know in which states $p_1, \ldots, p_m$ the relabeling N arrives after processing the trees which will be substituted for the parameters $y_1, \ldots, y_m$, respectively, then $\phi_i(q)(p_1, \ldots, p_k)$ is the state in which N arrives after processing the part of the output tree of M that corresponds to $M_q(s_i)$. The information on $p_1, \ldots, p_k$ is encoded into the states of M'; i.e., each state of M' is of the form (q, φ), where $q \in Q^{(m)}$, $\varphi : [m] \to Q_N$, and $\varphi(j)$ is the state in Q_N in which N arrives after processing the tree which is substituted for y_j in a derivation by M. Together we have sufficient information to "run" N on the right-hand side of M to obtain the corresponding rules of M'. □

In the next lemma we will show that the n-fold application of rub_0 can be simulated by a single application of $\mathrm{rub}_{0,1}$; i.e., if we know that $\mathrm{rub}_{0,1}(L) \in yMTT(\mathcal{L})$, then this means that also $\mathrm{rub}_0^n(L)$ for any $n \geq 2$ is in $yMTT(\mathcal{L})$. Note that $\mathrm{rub}_0^n(L) = \mathrm{rub}_{b_1,\ldots,b_n}(L)$, where $b_1, \ldots, b_n$ are new symbols not in L.

Lemma 8. *Let $\mathcal{L}$ be a class of tree languages which is closed under finite state relabelings. If $\mathrm{rub}_{0,1}(L) \in yMTT(\mathcal{L})$ then for every $n \geq 2$, $\mathrm{rub}_{b_1,\ldots,b_n}(L) \in yMTT(\mathcal{L})$.*

Proof. It is straightforward to construct an MTT M_{yield} which translates every input tree into its yield, represented as a monadic tree (e.g., $\sigma(a, b)$ is translated into $a(\hat{b})$). In fact in Example 1(6, yield) of [BE98] it is shown that this tree translation can be defined in monadic second order logic (MSO). By Theorem 7.1 of [EM98] the MSO definable tree translations are precisely those realized by finite copying macro tree transducers. We will now define a top-down tree transducer M_n which translates a monadic tree over the ranked alphabet $\Sigma = \{0^{(1)}, 1^{(1)}, \hat{0}^{(0)}, \hat{1}^{(0)}\}$ into a tree with yield in $\{b_1, \ldots, b_n\}^*$. This is done as follows. We use a Huffman code to represent each b_i by a string over $\{0, 1\}$; more precisely, the string $0^i 1$ represents b_{i+1} for every $0 \leq i \leq n-1$. M_n has states $1, \ldots, n$ and, starting in state 1, it arrives in state i after processing $i-1$ consecutive 0s. In state i, M_n outputs b_i (in the yield) if it processes a 1 and moves back to state 1.

Let $n \geq 2$ and define $M_n = ([n], \Sigma, \Gamma, 1, R)$ to be the top-down tree transducer with $\Gamma = \{\gamma^{(2)}, b_1^{(0)}, \ldots, b_n^{(0)}\}$ and R as follows.

$$\begin{array}{lll}
\langle \nu, 1(x_1) \rangle & \to \gamma(b_\nu, \langle 1, x_1 \rangle) & \text{for } \nu \in [n] \\
\langle \nu, 0(x_1) \rangle & \to \langle \nu + 1, x_1 \rangle & \text{for } \nu \in [n-1] \\
\langle n, 0(x_1) \rangle & \to \gamma(b_n, \langle 1, x_1 \rangle) & \\
\langle \nu, \hat{1} \rangle & \to e & \text{for } \nu \in [n] \\
\langle \nu, \hat{0} \rangle & \to e & \text{for } \nu \in [n]
\end{array}$$

Clearly, $y\tau_{M_n}(T_\Sigma) = \{b_1, \dots, b_n\}^*$ and hence if $yL = \{0,1\}^*$ then

$$y\tau_{M_n}(\tau_{M_{\mathrm{yield}}}(L)) = \{b_1, \dots, b_n\}^*.$$

Let Δ be a ranked alphabet. If we change M_n to have as input ranked alphabet $\Sigma' = \Sigma \cup \{\delta^{(1)} \mid \delta \in \Delta^{(0)}\} \cup \{\hat{\delta}^{(0)} \mid \delta \in \Delta^{(0)}\}$, as output alphabet $\Gamma' = \Gamma \cup \Delta^{(0)}$, and for every $\nu \in [n]$ the additional rules $\langle \nu, \delta(x_1)\rangle \to \gamma(\delta, \langle 1, x_1\rangle)$ and $\langle \nu, \hat{\delta}\rangle \to \delta$, then for every tree language K over Δ with $yK = \mathrm{rub}_{0,1}(L)$, $y\tau_{M_n}(\tau_{M_{\mathrm{yield}}}(K)) = \mathrm{rub}_{b_1,\dots,b_n}(L)$.

We can now compose $\tau_{M_{\mathrm{yield}}}$ with τ_{M_n} to obtain again a finite copying MTT which realizes $\tau_{M_{\mathrm{yield}}} \circ \tau_{M_n}$. This follows from the fact that MSO definable translations are closed under composition (cf. Proposition 2 of [BE98]) and that M_n is finite copying (it is even linear, i.e., 1-copying).

In Corollary 7.9 of [EM98] it is shown that finite copying MTTs with regular look-ahead have the same string generating power as finite copying top-down tree transducers with regular look-ahead. Hence, there is a finite copying top-down tree transducer with regular look-ahead M'' such that $y\tau_{M''}(K) = \mathrm{rub}_{b_1,\dots,b_n}(L)$ if $yK = \mathrm{rub}_{0,1}(L)$. Since regular look-ahead can be simulated by a relabeling (see Proposition 18.1 in [GS97]) we get that $\mathrm{rub}_{b_1,\dots,b_n}(L) \in yT(DQRELAB(MTT(\mathcal{L})))$ and, by Lemma 7 and the closure of MTT under right composition with T (Theorem 4.12 of [EV85]), this means that $\mathrm{rub}_{b_1,\dots,b_n}(L)$ is in $yMTT(\mathcal{L})$. □

The proof of Lemma 8 in fact shows that $yMTT(\mathcal{L})$ is closed under deterministic generalized sequential machine (GSM) mappings. For the case of nondeterministic MTTs it is shown in Theorem 6.3 of [DE98] that the class of string languages generated by them is closed under nondeterministic GSM mappings.

We are now ready to prove that there is a string language which can be generated by a nondeterministic top-down tree transducer with monadic input but not by the composition closure of MTTs.

Theorem 9. $yN\text{-}T_{\mathrm{mon}}(REGT) - \bigcup_{n\geq 0} yMTT^n(REGT) \neq \varnothing$.

Proof. Let $n \geq 1$. Since $MTT^n(REGT)$ is closed (i) under intersection with $REGT$ (follows trivially from the fact that $REGT$ is preserved by the inverse of MTT^n, cf. Theorem 7.4(1) in [EV85]) and (ii) under finite state relabelings (Lemma 7), we can apply Theorem 6 to $\mathcal{L} = MTT^n(REGT)$. We obtain that $\mathrm{rub}_{b_1,\dots,b_n}(L) \in yMTT^n(REGT)$ implies $\mathrm{rub}_{b_1,\dots,b_{n-1}}(L) \in yMTT_{\mathrm{sp}}(MTT^{n-1}(REGT))$. By Theorem 5 the latter class equals $yT(MTT^{n-1}(REGT))$ and since $MTT \circ T = MTT$ (Theorem 4.12 of [EV85]), it equals $yMTT^{n-1}(REGT)$. Hence, by induction, $L \in yT(REGT)$.

Let us now consider the concrete language $L_{\mathrm{exp}} = \{a^{2^n} \mid n \geq 0\}$. By the above we know that if $\mathrm{rub}_{b_1,\dots,b_n}(L) \in yMTT^n(REGT)$, then $L \in yT(REGT)$. Hence for $L = \mathrm{rub}_b(L_{\mathrm{exp}})$ we get that $\mathrm{rub}_{b_1,\dots,b_n}(L) \in yMTT^n(REGT)$ implies $\mathrm{rub}_b(L_{\mathrm{exp}}) \in yT(REGT)$. But by Corollary 3.2.16 of [ERS80] it is known that $\mathrm{rub}_b(L_{\mathrm{exp}})$ is *not* in $yT(REGT)$ (the proof uses a bridge theorem which would imply that L_{exp} can be generated by a finite copying top-down tree transducer;

but the languages generated by such transducers have the "Parikh property" and hence cannot be of exponential growth).

Altogether we get that $\mathrm{rub}_{b_1,\ldots,b_n,b}(L_{\mathrm{exp}})$ is not in $yMTT^n(REGT)$. By Lemma 8 this means that $\mathrm{rub}_{0,1}(L_{\mathrm{exp}}) \notin yMTT^n(REGT)$. It is easy to show that $\mathrm{rub}_{0,1}(L_{\mathrm{exp}})$ can be generated by a nondeterministic top-down tree transducer with monadic input; in fact, in Corollary 3.2.16 of [ERS80] it is shown that this language can be generated by an ET0L system. The class of languages generated by ET0L systems is precisely the class of string languages generated by nondeterministic top-down tree transducers with monadic input [Eng76]. □

Note that the last statement in the proof of Theorem 9 implies that $ET0L - \bigcup_{n\geq 0} yMTT^n(REGT) \neq \varnothing$, where $ET0L$ is the class of languages generated by ET0L systems. It is known that the IO-hierarchy $\bigcup_{n\geq 0} yYIELD^n(REGT)$ is inside $\bigcup_{n\geq 0} yMTT^n(REGT)$ (this follows, e.g., from Corollary 4.13 of [EV85]). From Theorem 9 we obtain the following corollary.

Corollary 10. $\mathrm{rub}_{0,1}(L_{\mathrm{exp}})$ *is not in the IO-hierarchy.*

5 Conclusions and Further Research Topics

In this paper we have proved that macro tree transducers which are simple in the parameters generate the same class of string languages as top-down tree transducers. Furthermore we have shown that there is a string language which can be generated by a nondeterministic top-down tree transducer with a regular monadic input language but not by the composition closure of MTT.

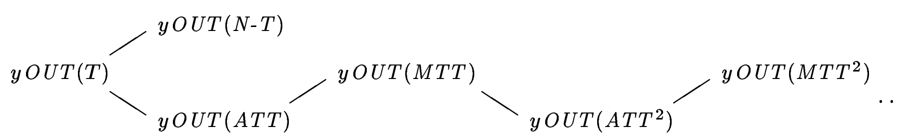

Fig. 2. Inclusion diagram for classes of string languages generated by tree transducers

Let us now consider another type of tree transducer: the attributed tree transducer (ATT) [Fül81]. Since the class ATT of translations realized by ATTs is a proper subclass of MTT it follows that $\mathrm{rub}_{0,1}(L_{\mathrm{exp}})$ is not in the class $yOUT(ATT)$ of string languages generated by ATTs. Since nondeterministic top-down tree transducers with monadic input equal cooperating regular tree grammars [FM98] and attributed tree transducers have the same term generating power as context-free hypergraph grammars, it follows that there is a tree language which can be generated by a cooperating regular tree grammar but not by a context-free hypergraph grammar. This remained open in [Man98a].

It is known that the class of string languages generated by top-down tree transducers is properly contained in that generated by ATTs (see, e.g., [Eng86]). Together with Theorem 9 this means that the two leftmost inclusions in Fig. 2 are proper (inclusions are edges going from left to right). However, it is open whether the other inclusions in Fig. 2 are proper. For instance, we do not know whether there is a language which can be generated by an MTT but not by an ATT. Note that for the corresponding classes of tree languages we know the answer: the language $\{\gamma^{2^n}(\alpha) \mid n \geq 0\}$ of monadic trees of exponential height can be generated by an MTT but not by an ATT (cf. Example 6.1 in [Man98b]).

Acknowledgement I wish to thank Joost Engelfriet for helpful discussions.

References

[BE98] R. Bloem and J. Engelfriet. A comparison of tree transductions defined by monadic second order logic and by attribute grammars. Technical Report 98-02, Leiden University, 1998.

[CF82] B. Courcelle and P. Franchi-Zannettacci. Attribute grammars and recursive program schemes. *Theoret. Comput. Sci.*, 17:163–191 and 235–257, 1982.

[DE98] Frank Drewes and Joost Engelfriet. Decidability of finiteness of ranges of tree transductions. *Inform. and Comput.*, 145:1–50, 1998.

[EM98] J. Engelfriet and S. Maneth. Macro tree transducers, attribute grammars, and MSO definable tree translations. Technical Report 98-09, Leiden University, 1998.

[Eng76] J. Engelfriet. Surface tree languages and parallel derivation trees. *Theoret. Comput. Sci.*, 2:9–27, 1976.

[Eng80] J. Engelfriet. Some open questions and recent results on tree transducers and tree languages. In R.V. Book, editor, *Formal language theory; perspectives and open problems.* New York, Academic Press, 1980.

[Eng86] J. Engelfriet. The complexity of languages generated by attribute grammars. *SIAM J. Comput.*, 15(1):70–86, 1986.

[ERS80] J. Engelfriet, G. Rozenberg, and G. Slutzki. Tree transducers, L systems, and two-way machines. *J. of Comp. Syst. Sci.*, 20:150–202, 1980.

[EV85] J. Engelfriet and H. Vogler. Macro tree transducers. *J. of Comp. Syst. Sci.*, 31:71–146, 1985.

[Fis68] M.J. Fischer. *Grammars with macro-like productions.* PhD thesis, Harvard University, Massachusetts, 1968.

[FM98] Z. Fülöp and S. Maneth. A characterization of ET0L tree languages by cooperating regular tree grammars. (To appear in "Grammatical Models of Multi-Agent Systems", Gordon and Breach, London), 1998.

[Fül81] Z. Fülöp. On attributed tree transducers. *Acta Cybernetica*, 5:261–279, 1981.

[FV98] Z. Fülöp and H. Vogler. *Syntax-Directed Semantics – Formal Models based on Tree Transducers.* EATCS Monographs on Theoretical Computer Science (W. Brauer, G. Rozenberg, A. Salomaa, eds.). Springer-Verlag, 1998.

[GS97] F. Gécseg and M. Steinby. Tree automata. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages, Volume 3*, chapter 1. Springer-Verlag, 1997.

[Küh98] A. Kühnemann. A pumping lemma for output languages of macro tree transducers. Technical Report TUD/FI95/08, Technical University Dresden, 1998. (also in *Proc. CAAP'96, LNCS 1059*, pages 44-58. Springer-Verlag, 1997.).

[Man98a] S. Maneth. Cooperating distributed hyperedge replacement grammars. In A. Kelemenová, editor, *Proc. MFCS'98 Satellite Workshop on Grammar Systems*, pages 149–164. Silesian University, 1998. (To appear in *Grammars*).

[Man98b] S. Maneth. The generating power of total deterministic tree transducers. *Inform. and Comput.*, 147:111–144, 1998.